National Textile University

Department of Computer Science

Subject:

Operating System

Submitted to:

Sir Nasir Mehmood

Submitted by:

Eman Babar

Reg. number:

23-NTU-CS-FL-1148

Semester:

5^{th}- A

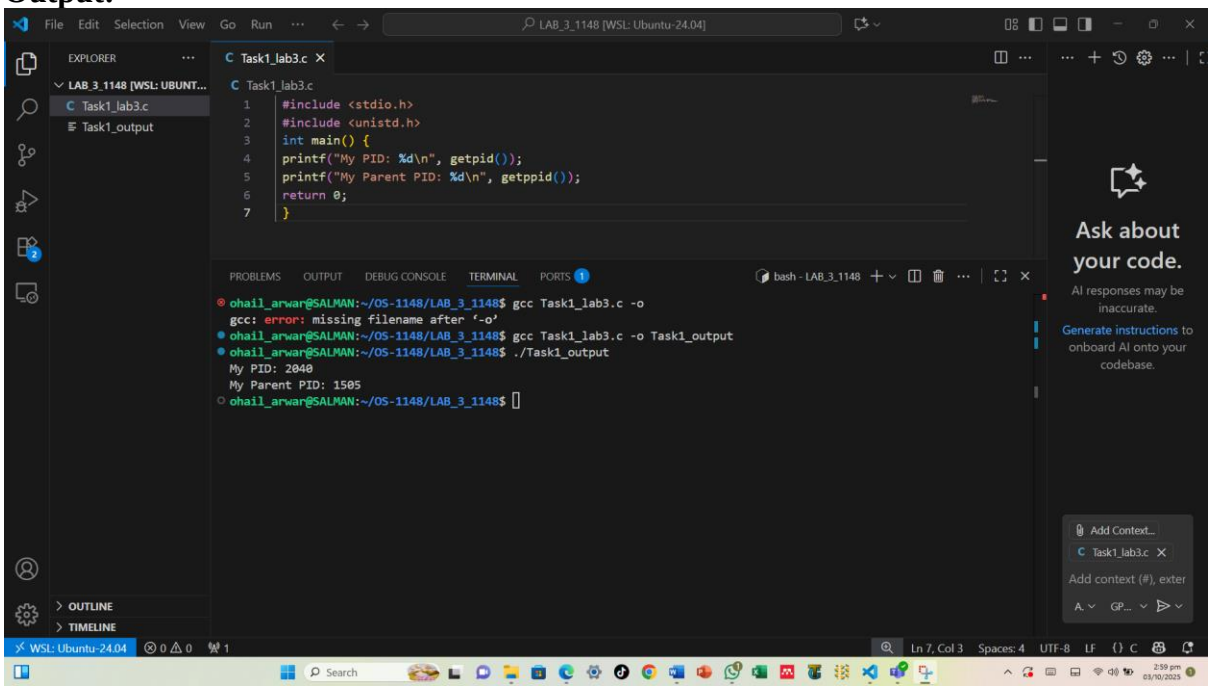
LAB-03

Program 1: Print PID and PPID

Code:

```
#include <stdio.h>
 #include <unistd.h>
 int main() {
 printf("My PID: %d\n", getpid());
 printf("My Parent PID: %d\n", getppid());
 return 0;
 }
```

Output:

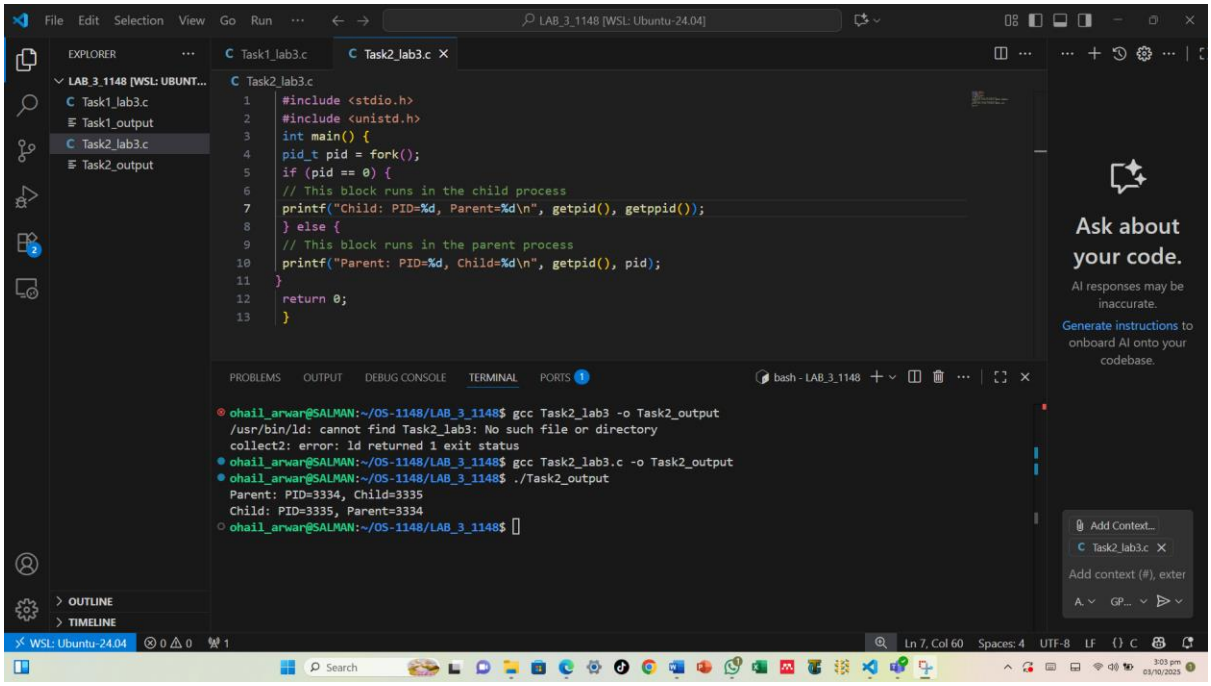


Program 2: Fork – Creating Child Process

Code:

```
#include <stdio.h>
 #include <unistd.h>
 int main() {
 pid_t pid = fork();
 if (pid == 0) {
 // This block runs in the child process
 printf("Child: PID=%d, Parent=%d\n", getpid(), getppid());
 } else {
 // This block runs in the parent process
 printf("Parent: PID=%d, Child=%d\n", getpid(), pid);
}
 return 0;
 }
```

Output:

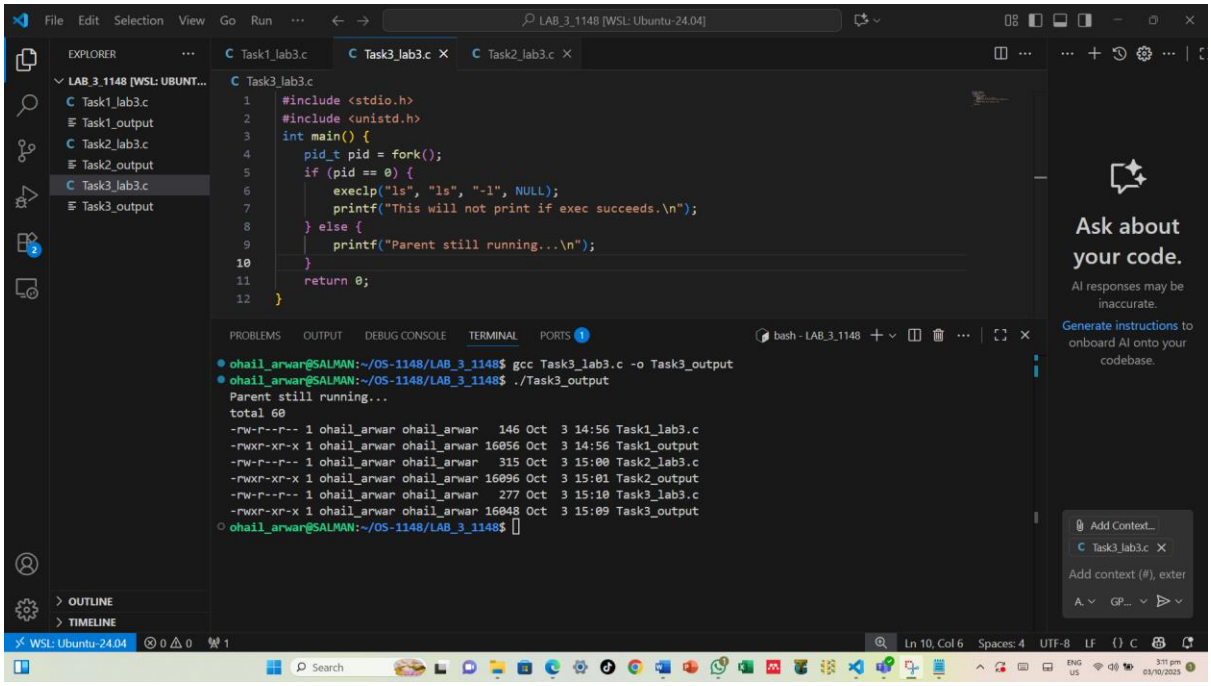


Program 3: Execl – Replacing a Process

Code:

```
#include <stdio.h>
 #include <unistd.h>
 int main() {
    pid_t pid = fork();
    if (pid == 0) {
        execlp("ls", "ls", "-l", NULL);
        printf("This will not print if exec succeeds.\n");
    } else {
        printf("Parent still running...\n");
    }
    return 0;
}
```

Output:

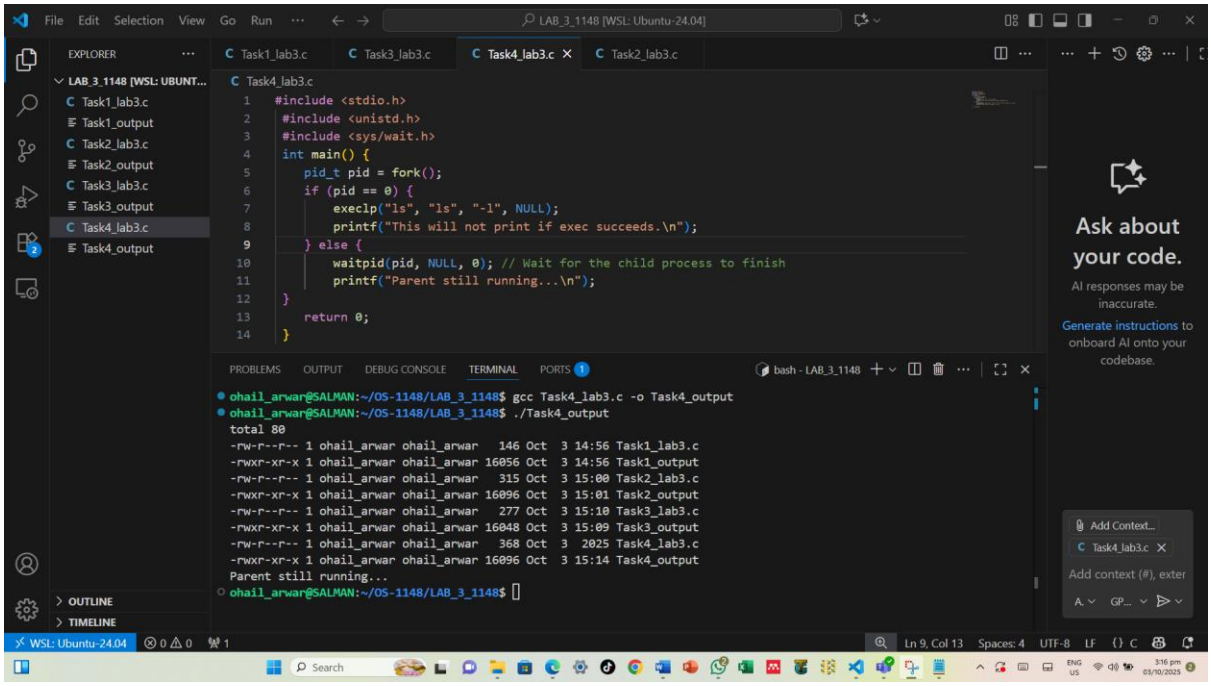


Program 4: Wait – Synchronization

Code:

```
#include <stdio.h>
#include <unistd.h>
#include <sys/wait.h>
int main() {
    pid_t pid = fork();
    if (pid == 0) {
        execlp("ls", "ls", "-l", NULL);
        printf("This will not print if exec succeeds.\n");
    } else {
        waitpid(pid, NULL, 0); // Wait for the child process to finish
        printf("Parent still running...\n");
 }
    return 0;
 }
```

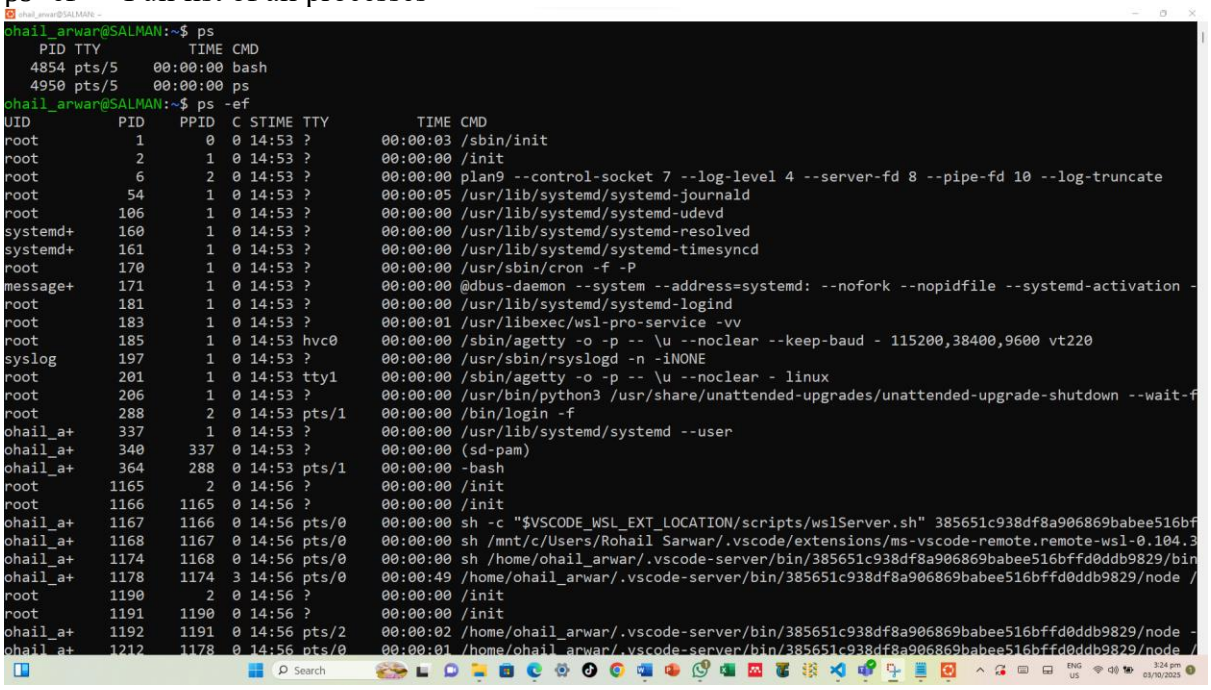
Output:



Linux Process Commands

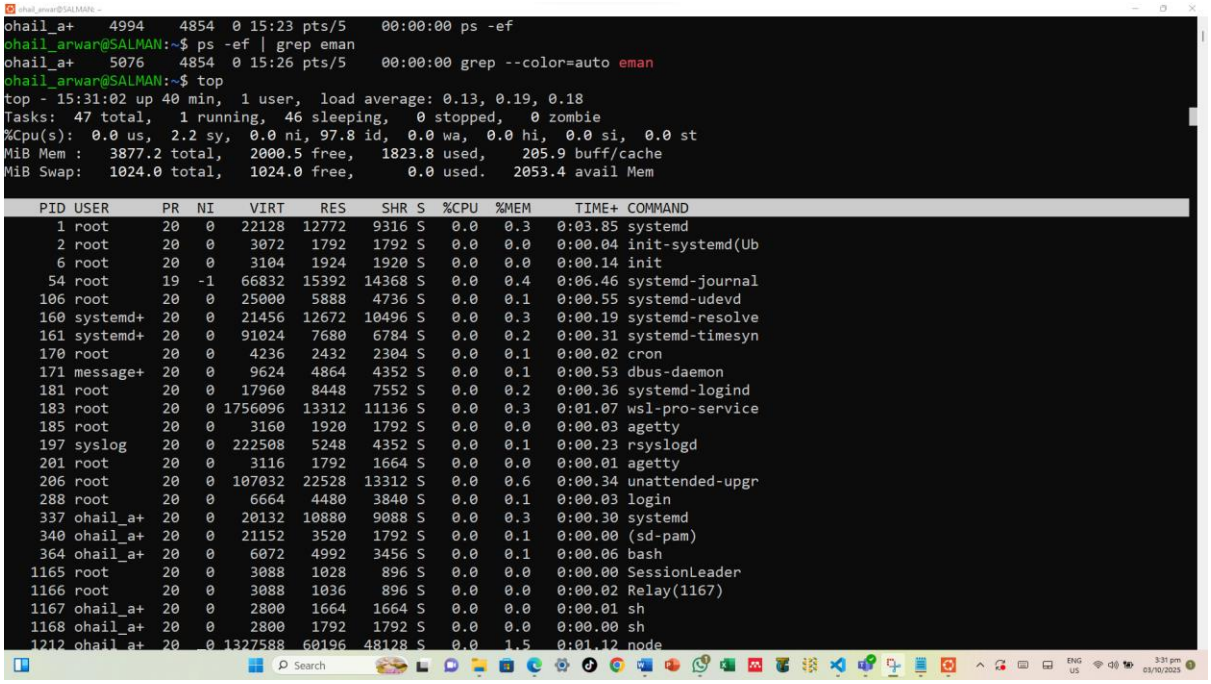
ps → Process Status

ps -ef → Full list of all processes

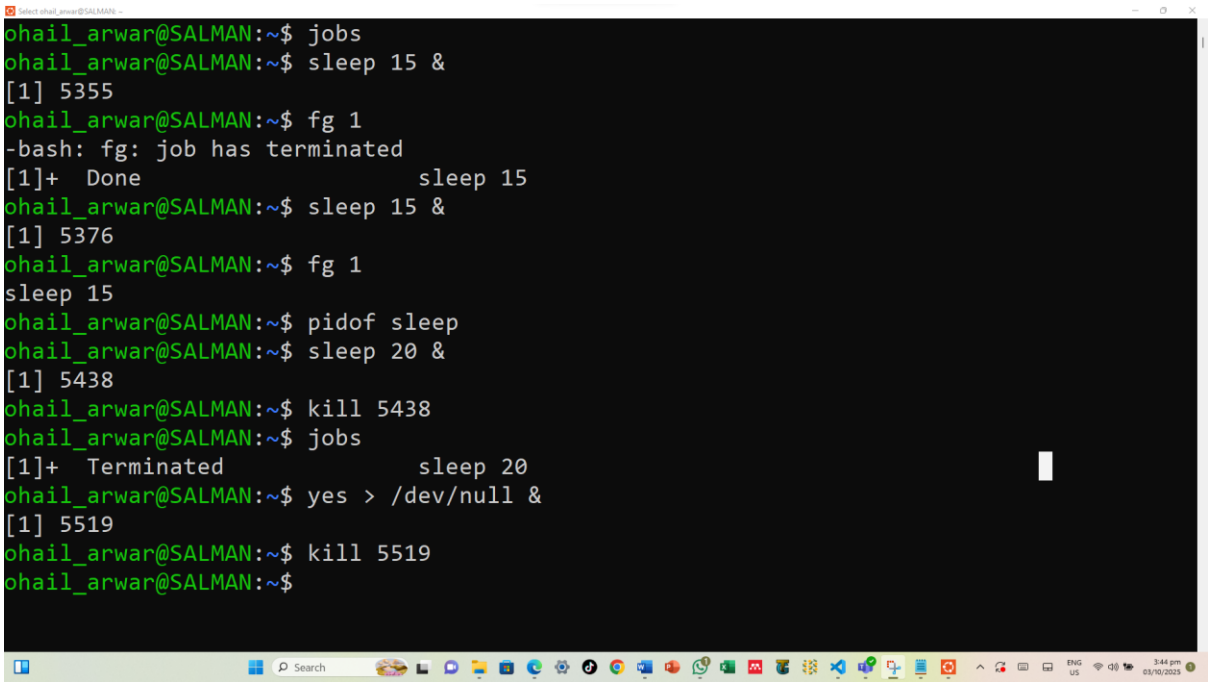


ps-ef | grep bash → This finds all processes related to the bash shell

top → Dynamic process viewer



Foreground:
A process that takes control of the terminal until it finishes.
sleep 30 → You cannot type new commands until it finishes.
Background:
Add & to run without blocking.
sleep 30 & → Terminal is free while the command runs.
Check background jobs:
Jobs
Bring a job to foreground:
fg %1
Get PID of a process by name:
pidof sleep
Run an infinite process:
yes > /dev/null &
Kill it with:
kill-9 <PID>
Output:


ohail_arwar@SALMAN:~$ jobs
ohail_arwar@SALMAN:~$ sleep 15 &
[1] 5355
ohail_arwar@SALMAN:~$ fg 1
-bash: fg: job has terminated
[1]+ Done   sleep 15
ohail_arwar@SALMAN:~$ sleep 15 &
[1] 5376
ohail_arwar@SALMAN:~$ fg 1
sleep 15
ohail_arwar@SALMAN:~$ pidof sleep
ohail_arwar@SALMAN:~$ sleep 20 &
[1] 5438
ohail_arwar@SALMAN:~$ kill 5438
ohail_arwar@SALMAN:~$ jobs
[1]+ Terminated   sleep 20
ohail_arwar@SALMAN:~$ yes > /dev/null &
[1] 5519
ohail_arwar@SALMAN:~$ kill 5519
ohail_arwar@SALMAN:~$
Search